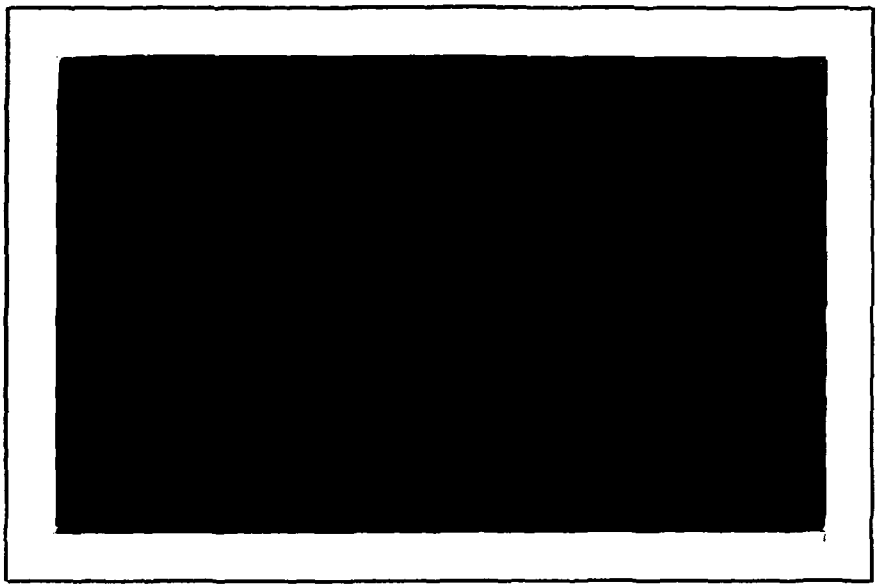
NAG 5-739

34P



IN-11630

UNIVERSITY OF HOUSTON

DEPARTMENT OF COMPUTER SCIENCE

Houston, Texas 77004

Semi Annual Report to
National Aeronautics and Space Administration

on

Database Interfaces on NASA's Heterogeneous Distributed Database System

Shou-Hsuan Stephen Huang
Department of Computer Science
University of Houston
Houston, Texas 77004

July 14, 1986 Grant NAG 5-739

Contents

1. Introduction

The purpose of ORACLE interface is to enable DAVID to submit queries and transactions to databases running under the ORACLE DBMS. The interface package is made up of several modules. We shall describe the progress of these modules below. In Section 2, we discuss the two approaches used in implementing the interface. Detail discussion of the design of the templates is shown in Section 3. Concluding remarks are given in Section 4.

2. Implementations of the interface

We take two different approaches in implementing the interface. The first method (called *HLI approach*) uses Host Language Interface facility of the ORACLE to retrieve data from ORACLE and calls the DAVID routines to insert the data into DAVID clusters. In order to achieve this, we have to derive a program (in host language such as Pascal) from a given primitive operation. To make the generation of such a program easier, a template is used for each primitive operation. We shall discuss the design of the templates in more detail in the next section.

Currently, Pascal templates for Selection-Projection, Selection-Multiprojection have been written. Programs to "fill" the Selection-Projection (Module 9.2.1.2.3) template are completed. We are working on the generalization of it to Selection-Multiprojection.

The second approach (called *Arbi Approach*) assumes that the resident DBMS does not provide an access method through the host language interface. So, we are using SQL (User-Friendly Interface of ORACLE) in the interface directly. After the result is retrieved from ORACLE, another program will reformat the result and store them into DAVID. Templates are used in this approach too. However, these

templates are in the form of system commands and SQL syntax, not in Pascal. The commands used in these templates are described in the next section. Currently, we have developed Module 9.2.1.2.1 (Oracle Define Cluster).

With these two approaches, we shall be able to know which approach is more efficient in terms of time and storage.

3. Design of the HLI Templates

We shall discuss the design of the templates used in the interface here. The template is either a Pascal program with some special commands embedded in it, or a collection of ORACLE statements with additional control commands. The interface module will take a template and convert it into a real Pascal program that can be compiled and run (or a system command file that can be executed). The template certainly depends on the operation to be performed on the database. It should be independent of most other factors (e. g., user-id, password, Boolean conditions, number of tables in a cluster). Thus we need the following three kinds of commands in the template.

[1] Repeating Commands: Some portion of the template may have to be repeated several times in the final output. For example, we have to bind several result variables. Although the syntax of such bind statement in Pascal is the same but we do not know how many columns there are in the cluster. Thus we cannot put several bind statements in the original template. So we put only one of them and add commands that indicate that the particular statement has to be repeated for as many times as there are columns. For each substituting command, there is an associated *index*. The use of the index will be explained later. There are several situations that such repeating occur. They are explained below.

```
Syntax   @begin<name>[_<separator>]
         @end<name>

<separator> ::=  c | s        (* comma or semicolon *)
<name> ::= (see definition below)
```

Meaning: Repeat the string between begin and end several times separated by <separator>. Default separator is blank. Append sequence number to all substituting commands and replace all indexes in the group.

<name>	Meaning (Repeat ..... times)
Rattr	Number of result attributes in a table
Rtable	Number of result tables in a cluster
Rcluster	Number of result clusters
Sattr	Number of source attributes
Stable	Number of source tables in a cluster
Scluster	Number of source clusters

Remark: This is not a complete list of the commands. More commands may be added as needed.

If there is a substituing command (say, @xxx) in the string to be repeated, then the command will be appended by a sequence number such that the command plus the sequence number (@xxx-i) is unique. This unique identification allows us to perform the substitution later. The details are given in the next subsection.

If there is an index command of this repeating command (say, [@1]) in the string to be repeated, then the command will be replaced by the index of the repeating command. Notice that this index may not be unique since we may have two identical indexes in the repeating group.

Sometimes when repeating a statement, these statements have to be separated

by a non-blank character (such as ',' or ';') for syntactical reasons. However, if a statement is repeated n times, there should be only n-1 such separators. Thus we may add such a separator at the end of a command such as "Sattr_s". For example, if we are repeating an attribute within a DAVID table definition, we have to separate these attributes by ",".

```
Template:
  .....
  table @tablename
        ( @beginrattr_c
              @rattr @rtype ( @rlength )
          @endrattr )
  .....

Template after Repeating:
  .....
  table @tablename
        ( @rattr-1 @rtype-1 ( @rlength-1 ) ,
          @rattr-2 @rtype-2 ( @rlength-2 ) ,
          @rattr-3 @rtype-3 ( @rlength-3 ) ,
          @rattr-4 @rtype-4 ( @rlength-4 ) ,
          @rattr-5 @rtype-5 ( @rlength-5 )
        )

Template after Substitution:
  .....
  table student
       (semester char(6),
        year     dec(4),
        dept     char(4),
        studid   char(9),
        studname char(20))
```

In the example above, we simplified it by assuming there can be one table per cluster definition. In the actual template, the table definition is within another pair of begin-end.

[2] Substituting Commands: These commands substitute names by their proper contents derived from next_gsql. For example, user-id of ORACLE only need to be filled in once. If these names appear in a repeating group (described above), then they have are indexed by 1, 2, ..., n so that the content can be uniquely identified. In the last example, since @rattr is in a repeating group that should be repeated 5 times (once for each variable), thus they are indexed as @rattr-1, @rattr-2, @rattr-3, @rattr-4 and @rattr-5. Then we can find the five attributes from next-gsql and replace the five attributes correctly.

Syntax ::= @<name>[-i]

Name	Meaning
Dpassword	DAVID Password
Duserid	DAVID User id
Opassword	ORACLE password
Ouserid	Oracle user id
Oquery	ORACLE SQL query
Rattr	name of an attribute in a table
Rcluster_hdg	name of a DAVID cluster
Rclusterdef	DAVID result cluster definition
Rclustername	DAVID result cluster name
Rlength	length of an attribute
Rprec	precision of variable
Rtablename	name of a table in a cluster
Rtablenames	names of all tables
Rtype	type of an attribute in a table
Sattr	name of source attributes
Sdbname	name of ORACLE database
Stype	Oracle source variable type
Tempfile	OR + last 5 characters of Command-ID

Remark: This is not a complete list of the commands. More commands may be added as needed.

[3] Index: Sometimes in the repeating group, we have to use the index of the repeating loop. Each loop in the template has a default index [@i] where i is the level number with the level number of the outmost level equals to 1. Thus, to access the second level index, we can use [@2].

```
Template:
    .....
    @begincluster
        define[@1] := '@def';
    @endcluster

Template after Repeating:     (2 clusters)
    .....
        define1 := '@def-1';
        define2 := '@def-2';
    .....
```

```
Syntax ::= [@<index>]

<index> ::= 1 | 2 | ...  | 9

Meaning:  place an index corresponding to the i-th level
          of the begin-end repeating group.
```

Name	Meaning
[@n]	replace with index of the n-th level of repeating group

4. Concluding Remarks

Our main effort in the last few months was concentrated on the design of the templates. It turns out that the templates we designed are fairly powerful. Our goals are:

- Template commands can be used in both approaches (HLI and Arbi);
- Template commands are independent of (host) the programming languages;
- Template commands are independent of resident DBMS's.
- The program to *fill* the templates is independent of the templates (i. e., independent of the primitives).

Modules 9.2.1.1.1 and 9.2.1.2.2 are working at this time. We are trying to get Module 9.2.1.2.3 working so that it can be demonstrated in September. They should be ready soon. For the rest of the year, we shall develop other modules in 9.2.1.2.

Appendix 1: Data Structures–Pascal

```
[Environment ('cdc')]

Module cdc (input, output);

Const
   type_length = 5;
   store_length = 5;
   command_id_length = 6;
   UidPwd_length = 10;
   field_length = 10;
   table_length = 10;
   cluster_length = 10;
   op_length = 10;
   db_length = 10;
   sys_config_length = 10;
   arbi_name_length = 12;
   resident_name_length = 15;
   max_result = 20;
   result_length = 100;
   clause_length = 100;
   select_length = 100;
   where_length = 500;

Type
   UidPwd_string      = packed array [1..uidpwd_length]      of char;
   Command_id_string  = packed array [1..command_id_length]  of char;
   arbi_name_string   = packed array [1..arbi_name_length]   of char;
   resident_name_string = packed array [1..resident_name_length] of char;
   type_string        = packed array [1..type_length]        of char;
   store_string       = packed array [1..store_length]       of char;
   field_string       = packed array [1..field_length]       of char;
   table_string       = packed array [1..table_length]       of char;
   cluster_string     = packed array [1..cluster_length]     of char;
   op_string          = packed array [1..op_length]          of char;
   db_string          = packed array [1..db_length]          of char;
   sys_config_string  = packed array [1..sys_config_length]  of char;
   clause_string      = packed array [1..clause_length]      of char;
   result_string      = packed array [1..result_length]      of char;
   select_string      = packed array [1..select_length]      of char;
   where_string       = packed array [1..where_length]       of char;

   Ptr_Vca = ^Vca_Record;

   Vca_Record = Record
         info : char;
         next : Ptr_Vca;
         end;

   Ptr_Field_Row = ^Field_Row;

   Field_Row = Record
         Field_Name     : field_string;
         Field_Type     : type_string;
         length,
         precision      : integer;
         Next_Field_Row : Ptr_Field_Row;
      End;

   Ptr_Table_Row = ^Table_Row;

   Table_Row = Record
         Table_Name      : table_string;
         Next_Table_Row  : Ptr_Table_Row;
         Child_Field_Row : Ptr_Field_Row;
      End;
```

```
   Ptr_Cluster_Row = ^Cluster_Row;

   Cluster_Row = Record
         Cluster_Name      : cluster_string;
         ResUid, ResPass1,
         ResPass2, ResPass3 : UidPwd_string;
         Arbi_File_Name     : arbi_name_string;
         Resident_Name      : resident_name_string;
         Next_Cluster_Row   : Ptr_Cluster_Row;
         Child_Table_Row    : Ptr_Table_Row;
      End;

   Gsql_Primitive_Type = Record
         Gsql_Result_Name : array [1..max_result] of result_string;
         Select_Clause    : array [1..max_result] of select_string;
         From1,
         From2            : clause_string;
         Where            : where_string;
         Store            : store_string;
      End;

   Ptr_Gsql_Row = ^Gsql_Row;

   Gsql_Row = Record
         vca                : Ptr_Vca;
         Command_id         : command_id_string;
         OpType             : op_string;
         DbType             : db_string;
         Sys_Config         : sys_config_string;
         Gsql_Primitive     : Gsql_Primitive_Type;
         Source_Cluster_Row,
         Result_Cluster_Row : Ptr_Cluster_Row;
      End;


   InsertType = Record
            Insert_Result,
            Insert_Select,
            Insert_Where,
            Insert_Values : clause_string;
         End;

   DeleteType = Record
            Delete_Result,
            Delete_Table,
            Delete_Item_Where : clause_string;
         End;

   UpdateType = Record
            Update_Result,
            Update_Set,
            Update_Table_Where,
            Update_Item_Where : clause_string;
         End;


Var
   Next_Gsql              : Gsql_Row;
   Batch_File, Store_File : TEXT;
End.
```

Appendix 2: Selection-Projection Template

```
Program HLI (input, output);
CONST
   EndOfTable = 4;
   { userid and password }
   UidPwd = '@UIDPWD';
   { DAVID uid and pwd }
   UID = '@UID';
   PASSWD = '@PASSWD';
   { DAVID result  name }
   @BEGINRESNAME RESULT@1 = '@RESULT'; @ENDRESNAME

TYPE
   string5 = packed array [1..5] of char;
   string20 = packed array [1..20] of char;
   string15  = packed array [1..15]  of char;
   string120 = packed array [1..120] of char;
   string400 = packed array [1..400] of char;
   Vstring100 = Varying [100] of char;
   HliVcaType = Record
                   code : integer;
                   others : string5;
                end;
   VcaType = string5;
   CcaType = string5;
   GcaType = string5;
   SccaType = string5;
   SourceType = Vstring100;
   ViewNameType = Vstring100;
   UidType = string5;
   PasswdType = string5;
   QueryType = string400;
   FlagType = string5;
   IdstringType = string5;
   TableNameType = string20;
   ColumnType = string20;
   ProgvarType = string20;
   ProgTypeType = integer;
   Ptr_Vca = ^VcaType;
   Ptr_Gca = ^GcaType;
   Ptr_Cca = ^CcaType;
   ptr_scca = ^Sccatype;
   word   = [word] -32768..32767;
   byte64 = array [1..32] of word;

VAR
   i, j, k : integer;
   { CURSOR Area and SQL Communications Area }
   CURS, SQLDCA: byte64;
   { DAVID Variables }
   hlivca : HliVcaType;
   vca : VcaType;
   pvca : Ptr_Vca;
   cca : CcaType;
   pcca : Ptr_Cca;
   gca : GcaType;
   pgca : Ptr_Gca;
   scca : SccaType;
   source : SourceType;
   ViewName : ViewNameType;
   flag : FlagType;
   idstring : IdstringType;
   tablename : TableNameType;
   column : ColumnType;
   progvar : ProgvarType;
   proglength : integer;
```

```
    { ORACLE query and Result Definition }
    query : QueryType;
    @BEGINDEFINITION define@1: QueryType;
    C@1 : ptr_cca;
    s@1 : ptr_scca;
    f@1 : TEXT; @ENDDEFINITION
    { Source and Result Variables }
    @BEGINSVAR s@SVAR : packed array [1..20] of char;
    r@SVAR : packed array [1..20] of char; @ENDSVAR

Procedure gsql_connect (uid : UidType; passwd : PasswdType;
                        pvca : Ptr_Vca; viewname : ViewNameType);
Begin
End;

Procedure gsql_run (vca : VcaType; query : QueryType;
                    flag : FlagType; source : sourcetype);
Begin
End;

Procedure gsql_compile (vca : VcaType; pgca : Ptr_Gca; query : QueryType;
                        flag : FlagType; idstring : IdstringType);
Begin
End;

Procedure gsql_eval (vca : VcaType; gca : GcaType;
                     flag : FlagType; idstring : IdstringType);
Begin
End;

Procedure asgcluster (vca : VcaType; pcca : Ptr_Cca;
                      source : SourceType; typ : char);
Begin
End;

Procedure bindcolumn (vca : VcaType; cca : CcaType; TableName : TableNameType;
                      column : ColumnType; progvar : ProgvarType;
                      progtype : integer; proglength : integer);
Begin
End;

Procedure asgsubcluster (vca : VcaType; cca : CcaType; scca : SccaType;
                         source : SourceType);
Begin
End;

Procedure scrinsert (vca : VcaType; pcca : ptr_cca; scca : ptr_Scca);
Begin
End;

Procedure deasgncluster (vca : VcaType; pcca : Ptr_Cca);
Begin
End;

Procedure gsqldrop (vca : VcaType; pgca : Ptr_Gca);
Begin
End;

Procedure logoff (uid : UidType; pvca : Ptr_Vca; viewname : ViewNameType);
Begin
End;

Procedure rollback (vca : VcaType);
Begin
End;
```

```
{ ORACLE Procedures }
Procedure Olon   (Var OSQLDCA : byte64;
                      OUId : string15;
                      OUIdLen : integer); EXTERN;
Procedure Oopen  (Var OCURS : byte64;
                  Var OSQLDCA : byte64); EXTERN;
Procedure Osql3  (Var OCURS : byte64;
                  Var Osqlstmt : string400;
                      OsqlLen : integer); EXTERN;
Procedure Odfinn (Var OCURS : byte64;
                      Oposition : integer;
                  Var Obuffer : string20;
                      Obuf1 : integer;
                      Oftype : integer); EXTERN;
Procedure Oexec  (Var OCURS : byte64); EXTERN;
Procedure Ofetch (Var OCURS : byte64); EXTERN;
Procedure Oclose (Var OCURS : byte64); EXTERN;
Procedure Ologof (Var OSQLDCA : byte64); EXTERN;
Procedure Oermsg (Var OCURS : integer;
                  Var Msgbuf : string120); EXTERN;


Procedure ErrorO (m, n : integer);
Var
   i : integer;
   msg : string120;
   err : TEXT;
Begin
   for i := 1 to 120 do
      msg [i] := ' ';
   Oermsg (n, msg);
   open (err, 'erro.dat', new);
   rewrite (err);
   Writeln (err, 'Error occurred during ORACLE ');
   case m of
     1 : writeln (err, 'OPEN ',msg);
     2 : writeln (err, 'LOGON ', msg);
     3 : writeln (err, 'SQL ', msg);
     4 : writeln (err, 'DEFINE ', msg);
     5 : writeln (err, 'FETCH ', msg);
     6 : writeln (err, 'EXECUTE ', msg);
   end;
   close (err);
End;


Procedure ErrorD (n : integer);
Var
   err : TEXT;
Begin
   write ('Error during DAVID ');
   Case n of
      1 : writeln (err, 'gsql conncet');
      2 : writeln (err, 'gsql run');
      3 : writeln (err, 'gsql compile');
      4 : writeln (err, 'gsql evaluate');
      5 : writeln (err, 'assign cluster');
      6 : writeln (err, 'bind column');
      7 : writeln (err, '');
      8 : writeln (err, 'assign subcluster');
      9 : writeln (err, '');
      10 : writeln (err, 'assign subcluster row');
      11 : writeln (err, 'deassign cluster');
      12 : writeln (err, 'gsql drop');
      13 : writeln (err, 'log  f');
```

```
   end;
end;

Procedure Build_Query;
Var
   i, j, k : integer;
   temp : string20;
Begin
   k := 1;
@BEGINSELECT
temp := '@SELECT';
for j := 1 to 20 do begin
Query [k] := temp [j];
k := k + 1;
end;
@ENDSELECT

@BEGINDEFINE
k := 1;
@BEGINDEF temp := '@DEF';
for j := 1 to 20 do begin
Define@1 [k] := temp [j];
k := k + 1;
end;
@ENDDEF
@ENDDEFINE

for i := 1 to 132 do
write (query [i]);
writeln;
End;


Procedure Convert (var f : text; t : string20);
var
   i, j, k, m, n, num : integer;
   need : boolean;
Begin
   i := 1;
   need := true;
   while (t [i] = ' ') do
      i := i + 1;
   while (i <= 20) and need do begin
      if not (t [i] in ['0'..'9']) then
         need := false;
      i := i + 1;
      end;
   i := 1;
   k := 0;
   num := 0;
   if need then begin
      while (t [i] = ' ') do i := i + 1;
      for j := 20 downto i do begin
         num := num + (ord (t [j]) - ord ('0')) * 10 ** k;
         k := k + 1;
         end;
      write (f, num : 6);
      end
   else begin
      m := 20;
      while (t [m] = ' ') do m := m - 1;
      write (f, '  ');
      if (m > 10) then
         for n := 1 to 20 do write (f, t [n])
      else
         for n := 1 to 10 do write (f, t [n]);
```

```
      end;
end;


Begin
  hlivca. code := 0;
  Build_Query;
  Olon (SQLDCA, UidPwd, 15);
  If (CURS [1] <> 0) Then ErrorO (1, CURS [1])
  Else Begin
    OOpen (CURS, SQLDCA);
    If (CURS [1] <> 0) Then ErrorO (2, CURS [1])
    Else Begin
      Osql3 (CURS, Query, 400);
      If (CURS [1] <> 0) Then ErrorO (3, CURS [1])
      Else Begin
        @BEGINODFINN ODFINN (CURS, @1, s@SVAR, 20, 1); @ENDODFINN
        If (CURS [1] <>0) Then ErrorO (4, CURS [1])
        Else Begin
          viewname := 'temp.pas';
          gsql_connect (uid, passwd, pvca, viewname);
          if (hlivca. code < 0) then errord (1)
          else begin
            @BEGINGSQL
            gsql_run (vca, DEFINE@1, flag, RESULT@1);
            if (hlivca. code < 0) then errord (2)
            else begin
               asgcluster (vca, c@1, RESULT@1, 'W');
               if (hlivca. code < 0) then errord (5)
               else begin
@BEGINBINDT tablename := '@TABLENAME';
      @BEGINBINDC
            column := '@COLUMN';
            bindcolumn (vca, cca, tablename, column, r@RVAR,
            @TYPE, @LENGTH);
            if (hlivca. code < 0) then errord (6)
            else begin
            @ENDBINDC @ENDBINDT
            gsqldrop (vca, pgca);
            if (hlivca. code < 0) then errord (12)
            else begin
            asgsubcluster (vca, cca, scca, RESULT@1);
            if (hlivca. code < 0) then errord (8)
            else begin
            open (f@1, 'f@1.dat', new); rewrite (f@1);
            @ENDGSQL
               OExec (CURS);
               If (CURS [1] <> 0) Then ErrorO (5, CURS [1])
               Else Begin
                  Repeat
                     OFetch (CURS);
                     if (CURS [1] <> EndofTable) then begin
                     @BEGINGSQL @BEGINBINDT @BEGINBINDC
                     r@RVAR := s@RVAR; convert (f@1, s@RVAR);
                     @ENDBINDC @ENDBINDT writeln (f@1);
                     scrinsert (vca, c@1, s@1);
                     if (hlivca. code < 0) then begin
                       rollback (vca);
                       errord (10);
                       end;
                     @ENDGSQL
                     end;
                  Until (CURS [1] <> 0) or (CURS [1] = EndOfTable)
                     or (hlivca. code < 0);
                  end;
@BEGINASGSUB
```

```
end;
@ENDASGSUB
@BEGINDEASG
close (f@1); end;
@BEGINBINDT  @BEGINBINDC
end; @ENDBINDT @ENDBINDC
end;
deasgncluster (vca, pcca);
if (hlivca. code < 0) then errord (11);
end;
@ENDDEASG
            end;
            logoff (uid, pvca, viewname);
            if (hlivca. code < 0) then errord (13);
            End;
         End;
      End;
   End;
   OClose (CURS);
   OLogof (SQLDCA);
End.
```

Appendix 3: *Filled* Selection-Projection Template

```
Program HLI (input, output);
CONST
   EndOfTable = 4;
   { userid and password }
   UidPwd = 'csgr7399/trywer';
   { DAVID uid and pwd }
   UID = 'joshu';
   PASSWD = 'settr';
   { DAVID result  name }
    RESULT1 = 'usr3.rcis';
 RESULT2 = 'usr3.teacher';

TYPE
   string5 = packed array [1..5] of char;
   string20 = packed array [1..20] of char;
   string15  = packed array [1..15]  of char;
   string120 = packed array [1..120] of char;
   string400 = packed array [1..400] of char;
   Vstring100 = Varying [100] of char;
   HliVcaType = Record
                   code : integer;
                   others : string5;
                end;
   VcaType = string5;
   CcaType = string5;
   GcaType = string5;
   SccaType = string5;
   SourceType = Vstring100;
   ViewNameType = Vstring100;
   UidType = string5;
   PasswdType = string5;
   QueryType = string400;
   FlagType = string5;
   IdstringType = string5;
   TableNameType = string20;
   ColumnType = string20;
   ProgvarType = string20;
   ProgTypeType = integer;
   Ptr_Vca = ^VcaType;
   Ptr_Gca = ^GcaType;
   Ptr_Cca = ^CcaType;
   ptr_scca = ^Sccatype;
   word   = [word] -32768..32767;
   byte64 = array [1..32] of word;

VAR
   i, j, k : integer;
   { CURSOR Area and SQL Communications Area }
   CURS, SQLDCA: byte64;
   { DAVID Variables }
   hlivca : HliVcaType;
   vca : VcaType;
   pvca : Ptr_Vca;
   cca : CcaType;
   pcca : Ptr_Cca;
   gca : GcaType;
   pgca : Ptr_Gca;
   scca : SccaType;
   source : SourceType;
   ViewName : ViewNameType;
   flag : FlagType;
   idstring : IdstringType;
   tablename : TableNameType;
   column : ColumnType;
   progvar : ProgvarType;
```

```
    proglength : integer;
    { ORACLE query and Result Definition }
    query : QueryType;
     define1: QueryType;
    C1 : ptr_cca;
    s1 : ptr_scca;
    f1 : TEXT;
  define2: QueryType;
    C2 : ptr_cca;
    s2 : ptr_scca;
    f2 : TEXT;
    { Source and Result Variables }
     ssemester : packed array [1..20] of char;
    rsemester : packed array [1..20] of char;
  syear : packed array [1..20] of char;
    ryear : packed array [1..20] of char;
  sdept : packed array [1..20] of char;
    rdept : packed array [1..20] of char;
  sstudname : packed array [1..20] of char;
    rstudname : packed array [1..20] of char;
  sstudid : packed array [1..20] of char;
    rstudid : packed array [1..20] of char;
  sgrade : packed array [1..20] of char;
    rgrade : packed array [1..20] of char;
  scoursenum : packed array [1..20] of char;
    rcoursenum : packed array [1..20] of char;
  ssecnum : packed array [1..20] of char;
    rsecnum : packed array [1..20] of char;
  sinstname : packed array [1..20] of char;
    rinstname : packed array [1..20] of char;
  sinstid : packed array [1..20] of char;
    rinstid : packed array [1..20] of char;

Procedure gsql_connect (uid : UidType; passwd : PasswdType;
                        pvca : Ptr_Vca; viewname : ViewNameType);
Begin
End;

Procedure gsql_run (vca : VcaType; query : QueryType;
                    flag : FlagType; source : sourcetype);
Begin
End;

Procedure gsql_compile (vca : VcaType; pgca : Ptr_Gca; query : QueryType;
                        flag : FlagType; idstring : IdstringType);
Begin
End;

Procedure gsql_eval (vca : VcaType; gca : GcaType;
                     flag : FlagType; idstring : IdstringType);
Begin
End;

Procedure asgcluster (vca : VcaType; pcca : Ptr_Cca;
                      source : SourceType; typ : char);
Begin
End;

Procedure bindcolumn (vca : VcaType; cca : CcaType; TableName : TableNameType;
                      column : ColumnType; progvar : ProgvarType;
                      progtype : integer; proglength : integer);
Begin
End;

Procedure asgsubcluster (vca : VcaType; cca : CcaType; scca : SccaType;
                         source : SourceType);
```

```
Begin
End;

Procedure scrinsert (vca : VcaType; pcca : ptr_cca; scca : ptr_Scca);
Begin
End;

Procedure deasgncluster (vca : VcaType; pcca : Ptr_Cca);
Begin
End;

Procedure gsqldrop (vca : VcaType; pgca : Ptr_Gca);
Begin
End;

Procedure logoff (uid : UidType; pvca : Ptr_Vca; viewname : ViewNameType);
Begin
End;

Procedure rollback (vca : VcaType);
Begin
End;


{ ORACLE Procedures }
Procedure Olon   (Var OSQLDCA : byte64;
                      OUId : string15;
                      OUIdLen : integer); EXTERN;
Procedure Oopen  (Var OCURS : byte64;
                  Var OSQLDCA : byte64); EXTERN;
Procedure Osql3  (Var OCURS : byte64;
                  Var Osqlstmt : string400;
                      OsqlLen : integer); EXTERN;
Procedure Odfinn (Var OCURS : byte64;
                      Oposition : integer;
                  Var Obuffer : string20;
                      Obufl : integer;
                      Oftype : integer); EXTERN;
Procedure Oexec  (Var OCURS : byte64); EXTERN;
Procedure Ofetch (Var OCURS : byte64); EXTERN;
Procedure Oclose (Var OCURS : byte64); EXTERN;
Procedure Ologof (Var OSQLDCA : byte64); EXTERN;
Procedure Oermsg (Var OCURS : integer;
                  Var Msgbuf : string120); EXTERN;


Procedure ErrorO (m, n : integer);
Var
   i : integer;
   msg : string120;
   err : TEXT;
Begin
   for i := 1 to 120 do
      msg [i] := ' ';
   Oermsg (n, msg);
   open (err, 'erro.dat', new);
   rewrite (err);
   Writeln (err, 'Error occurred during ORACLE ');
   case m of
     1 : writeln (err, 'OPEN ',msg);
     2 : writeln (err, 'LOGON ', msg);
     3 : writeln (err, 'SQL ', msg);
     4 : writeln (err, 'DEFINE ', msg);
     5 : writeln (err, 'FETCH ', msg);
     6 : writeln (err, 'EXECUTE ', msg);
   end;
```

```
   close (err);
End;


Procedure ErrorD (n : integer);
Var
   err : TEXT;
Begin
   write ('Error during DAVID ');
   Case n of
      1 : writeln (err, 'gsql conncet');
      2 : writeln (err, 'gsql run');
      3 : writeln (err, 'gsql compile');
      4 : writeln (err, 'gsql evaluate');
      5 : writeln (err, 'assign cluster');
      6 : writeln (err, 'bind column');
      7 : writeln (err, '');
      8 : writeln (err, 'assign subcluster');
      9 : writeln (err, '');
      10 : writeln (err, 'assign subcluster row');
      11 : writeln (err, 'deassign cluster');
      12 : writeln (err, 'gsql drop');
      13 : writeln (err, 'log off');
   end;
end;

Procedure Build_Query;
Var
   i, j, k : integer;
   temp : string20;
Begin
   k := 1;

temp := 'select semester ,yea';
for j := 1 to 20 do begin
Query [k] := temp [j];
k := k + 1;
end;


temp := 'r ,dept ,studname ,s';
for j := 1 to 20 do begin
Query [k] := temp [j];
k := k + 1;
end;


temp := 'tudid ,grade ,course';
for j := 1 to 20 do begin
Query [k] := temp [j];
k := k + 1;
end;


temp := 'num ,secnum ,instnam';
for j := 1 to 20 do begin
Query [k] := temp [j];
k := k + 1;
end;


temp := 'e ,instid from stude';
for j := 1 to 20 do begin
Query [k] := temp [j];
k := k + 1;
end;
```

```
temp := 'nt where year = 1979';
for j := 1 to 20 do begin
Query [k] := temp [j];
k := k + 1;
end;


temp := ' or dept=''cosc''';
for j := 1 to 20 do begin
Query [k] := temp [j];
k := k + 1;
end;


temp := '';
for j := 1 to 20 do begin
Query [k] := temp [j];
k := k + 1;
end;



k := 1;
 temp := 'define cluster usr3.';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;

 temp := 'rcis ( table reg ( s';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;

 temp := 'emester char ( 6), y';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;

 temp := 'ear dec ( 4), dept c';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;

 temp := 'har ( 4), student ta';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;

 temp := 'ble course table ),';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;

 temp := 'table student ( stud';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;
```

```
 temp := 'name char ( 10), stu';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;

 temp := 'did dec ( 4), grade';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;

 temp := 'char ( 1) ), table c';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;

 temp := 'ourse ( coursenum de';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;

 temp := 'c ( 3), secnum dec (';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;

 temp := '3) ) );';
for j := 1 to 20 do begin
Define1 [k] := temp [j];
k := k + 1;
end;



k := 1;
 temp := 'define cluster usr3.';
for j := 1 to 20 do begin
Define2 [k] := temp [j];
k := k + 1;
end;

 temp := 'teach( table instruc';
for j := 1 to 20 do begin
Define2 [k] := temp [j];
k := k + 1;
end;

 temp := 't ( dept char ( 4),';
for j := 1 to 20 do begin
Define2 [k] := temp [j];
k := k + 1;
end;

 temp := 'instname char ( 10),';
for j := 1 to 20 do begin
Define2 [k] := temp [j];
k := k + 1;
end;

 temp := ' instid dec ( 4) ) )';
for j := 1 to 20 do begin
```

```
Define2 [k] := temp [j];
k := k + 1;
end;

 temp := ';';
for j := 1 to 20 do begin
Define2 [k] := temp [j];
k := k + 1;
end;



for i := 1 to 132 do
write (query [i]);
writeln;
End;


Procedure Convert (var f : text; t : string20);
var
   i, j, k, m, n, num : integer;
   need : boolean;
Begin
   i := 1;
   need := true;
   while (t [i] = ' ') do
      i := i + 1;
   while (i <= 20) and need do begin
      if not (t [i] in ['0'..'9']) then
         need := false;
      i := i + 1;
      end;
   i := 1;
   k := 0;
   num := 0;
   if need then begin
      while (t [i] = ' ') do i := i + 1;
      for j := 20 downto i do begin
         num := num + (ord (t [j]) - ord ('0')) * 10 ** k;
         k := k + 1;
         end;
      write (f, num : 6);
      end
   else begin
      m := 20;
      while (t [m] = ' ') do m := m - 1;
      write (f, '  ');
      if (m > 10) then
         for n := 1 to 20 do write (f, t [n])
      else
         for n := 1 to 10 do write (f, t [n]);
      end;
end;


Begin
  hlivca. code := 0;
  Build Query;
  Olon (SQLDCA, UidPwd, 15);
  If (CURS [1] <> 0) Then ErrorO (1, CURS [1])
  Else Begin
    OOpen (CURS, SQLDCA);
    If (CURS [1] <> 0) Then ErrorO (2, CURS [1])
    Else Begin
      Osql3 (CURS, Query, 400);
      If (CURS [1] <> 0) Then ErrorO (3, CURS [1])
```

```
      Else Begin
         ODFINN (CURS, 1, ssemester, 20, 1);
ODFINN (CURS, 2, syear, 20, 1);
ODFINN (CURS, 3, sdept, 20, 1);
ODFINN (CURS, 4, sstudname, 20, 1);
ODFINN (CURS, 5, sstudid, 20, 1);
ODFINN (CURS, 6, sgrade, 20, 1);
ODFINN (CURS, 7, scoursenum, 20, 1);
ODFINN (CURS, 8, ssecnum, 20, 1);
ODFINN (CURS, 9, sinstname, 20, 1);
ODFINN (CURS, 10, sinstid, 20, 1);
        If (CURS [1] <>0) Then ErrorO (4, CURS [1])
        Else Begin
          viewname := 'temp.pas';
          gsql_connect (uid, passwd, pvca, viewname);
          if (hlivca. code < 0) then errord (1)
          else begin

            gsql_run (vca, DEFINE1, flag, RESULT1);
            if (hlivca. code < 0) then errord (2)
            else begin
               asgcluster (vca, c1, RESULT1, 'W');
               if (hlivca. code < 0) then errord (5)
               else begin
tablename := 'reg';

            column := 'semester';
            bindcolumn (vca, cca, tablename, column, rsemester,
            1, 6);
            if (hlivca. code < 0) then errord (6)
            else begin


            column := 'year';
            bindcolumn (vca, cca, tablename, column, ryear,
            2, 4);
            if (hlivca. code < 0) then errord (6)
            else begin


            column := 'dept';
            bindcolumn (vca, cca, tablename, column, rdept,
            1, 4);
            if (hlivca. code < 0) then errord (6)
            else begin

tablename := 'student';

            column := 'studname';
            bindcolumn (vca, cca, tablename, column, rstudname,
            1, 10);
            if (hlivca. code < 0) then errord (6)
            else begin


            column := 'studid';
            bindcolumn (vca, cca, tablename, column, rstudid,
            2, 4);
            if (hlivca. code < 0) then errord (6)
            else begin


            column := 'grade';
            bindcolumn (vca, cca, tablename, column, rgrade,
            1, 1);
            if (hlivca. code < 0) then errord (6)
```

```
                else begin

        tablename := 'course';

                column := 'coursenum';
                bindcolumn (vca, cca, tablename, column, rcoursenum,
                2, 3);
                if (hlivca. code < 0) then errord (6)
                else begin


                column := 'secnum';
                bindcolumn (vca, cca, tablename, column, rsecnum,
                2, 3);
                if (hlivca. code < 0) then errord (6)
                else begin

                gsqldrop (vca, pgca);
                if (hlivca. code < 0) then errord (12)
                else begin
                asgsubcluster (vca, cca, scca, RESULT1);
                if (hlivca. code < 0) then errord (8)
                else begin
                open (f1, 'f1.dat', new); rewrite (f1);


                gsql_run (vca, DEFINE2, flag, RESULT2);
                if (hlivca. code < 0) then errord (2)
                else begin
                   asgcluster (vca, c2, RESULT2, 'W');
                   if (hlivca. code < 0) then errord (5)
                   else begin
        tablename := 'instruct';

                column := 'dept';
                bindcolumn (vca, cca, tablename, column, rdept,
                1, 4);
                if (hlivca. code < 0) then errord (6)
                else begin


                column := 'instname';
                bindcolumn (vca, cca, tablename, column, rinstname,
                1, 10);
                if (hlivca. code < 0) then errord (6)
                else begin


                column := 'instid';
                bindcolumn (vca, cca, tablename, column, rinstid,
                2, 4);
                if (hlivca. code < 0) then errord (6)
                else begin

                gsqldrop (vca, pgca);
                if (hlivca. code < 0) then errord (12)
                else begin
                asgsubcluster (vca, cca, scca, RESULT2);
                if (hlivca. code < 0) then errord (8)
                else begin
                open (f2, 'f2.dat', new); rewrite (f2);

                   OExec (CURS);
                   If (CURS [1] <> 0) Then ErrorO (5, CURS [1])
                   Else Begin
                      Repeat
```

```
                        OFetch (CURS);
                        if (CURS [1] <> EndofTable) then begin

                        rsemester := ssemester; convert (f1, ssemester);


                        ryear := syear; convert (f1, syear);


                        rdept := sdept; convert (f1, sdept);


                        rstudname := sstudname; convert (f1, sstudname);


                        rstudid := sstudid; convert (f1, sstudid);


                        rgrade := sgrade; convert (f1, sgrade);


                        rcoursenum := scoursenum; convert (f1, scoursenum);


                        rsecnum := ssecnum; convert (f1, ssecnum);
                          writeln (f1);
                        scrinsert (vca, c1, s1);
                        if (hlivca. code < 0) then begin
                          rollback (vca);
                          errord (10);
                          end;


                        rdept := sdept; convert (f2, sdept);


                        rinstname := sinstname; convert (f2, sinstname);


                        rinstid := sinstid; convert (f2, sinstid);
                          writeln (f2);
                        scrinsert (vca, c2, s2);
                        if (hlivca. code < 0) then begin
                          rollback (vca);
                          errord (10);
                          end;

                        end;
                      Until (CURS [1] <> 0) or (CURS [1] = EndOfTable)
                        or (hlivca. code < 0);
                      end;

    end;


    end;


    close (f1); end;

    end;

    end;

    end;
```

```
        end;

        end;

        end;

        end;

        end;
        end;
        deasgncluster (vca, pcca);
        if (hlivca. code < 0) then errord (11);
        end;


        close (f2); end;

        end;

        end;

        end;
        end;
        deasgncluster (vca, pcca);
        if (hlivca. code < 0) then errord (11);
        end;

                    end;
                    logoff (uid, pvca, viewname);
                    if (hlivca. code < 0) then errord (13);
                    End;
                End;
              End;
            End;
            OClose (CURS);
            OLogof (SQLDCA);
        End.
```

Appendix 4: Install Template

```
$ufi
   @UIDPWD
   spool @TEMPFILE
   select tname, cname, coltype, width from col
   where tname = ' ' @BEGINBINDT or tname = '@TABLENAME' @ENDBINDT ;
   exit
$run Exetract.exe
@TEMPFILE
```

Appendix 5: *Filled* Install Template

```
$ufi
   csgr7399/trywer
   spool ORabcde.lis
   select tname, cname, coltype, width from col
   where tname = ' '  or tname = 'STUDENT'
 or tname = 'INSTRUCT'  ;
   exit
$run extract.exe
ORabcde.lis
```